

IN THE CLAIMS:

The following listing of claims will replace all prior versions, and listings, of claims in the application.

Listing of Claims:

1. (Currently Amended) A method for manipulating a file in a filesystem, the filesystem having a first layer and a second layer, the method comprising:
if the file is in the first layer, retrieving the file from the first layer, and if the file is subsequently changed, saving at least a portion of the file in the first layer;
if the file is not in the first layer, retrieving the file from the second layer, and if the file is subsequently changed, saving at least a portion of the file in the first layer; and
if the file is not in the first layer or the second layer, creating the file and saving at least a portion of the file in the first layer.
2. (Original) The method of claim 1, wherein the portion includes attributes associated with the file.
3. (New) An overlay filesystem comprising:
a filesystem module residing in a kernel and configured to present a composite view of a plurality of layered directories such that all files are represented as residing in a topmost directory.

4. (New) The overlay filesystem of claim 3 wherein if two or more directories include a file with a same pathname then the file is represented as residing in a topmost directory.
5. (New) The overlay filesystem of claim 3 wherein a request to modify a file in an underlying directory is directed to the topmost directory.
6. (New) The overlay filesystem of claim 3 wherein the filesystem module is further configured to create an onode structure to represent an underlying directory node.
7. (New) An application processing network comprising:
 - an application capable of generating filesystem operations;
 - a vnode layer configured to receive filesystem operations from the application;
 - an overlay filesystem including
 - a back filesystem containing shared read-only files and
 - a front filesystem mounted above the back filesystem and containing writable files,
 - the overlay filesystem being configured to selectively route filesystem operations from the vnode layer to the front and back filesystems.

8. (New) The application processing network of claim 7 wherein a filesystem operation is an open() request for a file and the overlay filesystem is configured to allocate an onode including a shadow vnode, send the open() request to the front filesystem which returns a front vnode to be stored in the onode, and send the open() request to the back filesystem which returns a back vnode to be stored in the onode.
9. (New) The application processing network of claim 8 wherein the shadow vnode maintains a reference count that is incremented each time the file is opened.
10. (New) The application processing network of claim 8 wherein the onode and the shadow vnode are linked and shadow vnode is returned to the vnode layer.
11. (New) The application processing network of claim 10 wherein the vnode layer returns to the application a file descriptor linked to the shadow vnode.
12. (New) The application processing network of claim 7 wherein a filesystem operation is a read() request for a file and the overlay filesystem is configured to receive the read() request and a shadow vnode from the vnode layer, determine a vnode for a filesystem from an onode for the file, and pass the read() request to the filesystem.

13. (New) The application processing network of claim 12 wherein the vnode for the filesystem is a front vnode for the front filesystem.
14. (New) The application processing network of claim 7 wherein a structure of the vnode layer is filesystem-independent and the overlay filesystem maintains onodes.
15. (New) The application processing network of claim 14 wherein each onode can include a vnode triplet.
16. (New) The application processing network of claim 15 wherein the vnode triplet includes a shadow vnode pointer, a front vnode pointer, and a back vnode pointer that point to a shadow vnode, a front vnode, and a back vnode, respectively.
17. (New) The application processing network of claim 14 wherein each onode is stored in a hash table.
18. (New) The application processing network of claim 7 wherein the overlay filesystem includes more than two filesystems and is further configured to allocate and cache onodes.
19. (New) The application processing network of claim 7 wherein the overlay filesystem is further configured to support a snapshot/restore module.

20. (New) The application processing network of claim 7 wherein the overlay filesystem is further configured to implement a file in the front filesystem with a page-level copy-on-write structure.
21. (New) The application processing network of claim 20 wherein a format of the file in the front filesystem includes a header, a page map, and a page.
22. (New) The application processing network of claim 21 wherein the header stores extended file attributes, file verification data, virtual size, and reserved/padding.
23. (New) The application processing network of claim 21 wherein the page map includes a bitmap indicating a location of the page in the front filesystem.